matasano

Hardware Virtualization Rootkits

Dino A. Dai Zovi
ddz@matasano.com

Agenda

- Introductions
- Virtualization Overview
- Intel Virtual Machine Extensions
- Vitriol: The VT-x Rootkit
- Demonstration

Who We Are

Dave Goldsmith (@stake cofounder)

Jeremy Rauch (SecurityFocus cofounder)

Thomas Ptacek (Arbor)

Window Snyder (Microsoft XPSP2)

Dino Dai Zovi (Bloomberg)

What We Do

- DEPLOYSAFE
  Reverse and Pen-Test Products
  for enterprises
- SHIPSAFE
  Audit and Test Products
  for vendors
- CLOCKWORK
  our First Product
  coming July/August 2006

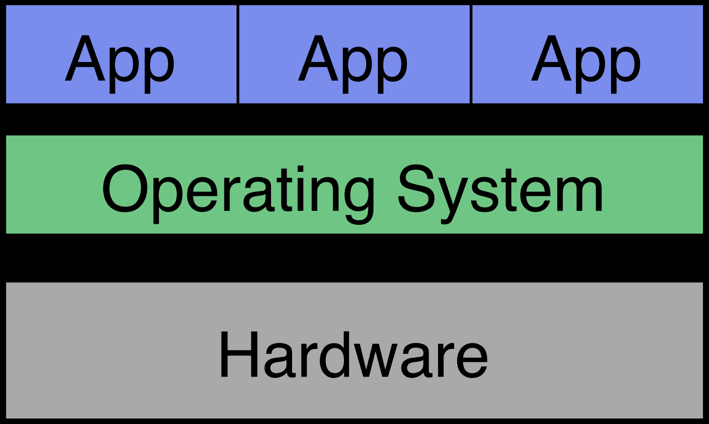
Why am I here?

- Most current CPUs now support Hardware Virtual Machines (HVMs)
- Virtualization, especially hardware-supported, offers tremendous space/power/cost savings to enterprises
- Hardware VM Rootkits run between the operating system and true hardware:
  - In memory pages inaccessible to the running operating system
  - Mediating access to devices, observing and filtering input/output
- HVM Rootkits can install themselves by migrating the running OS into a VM *while the OS is running*.

Overview of Virtualization

Traditional Operating System

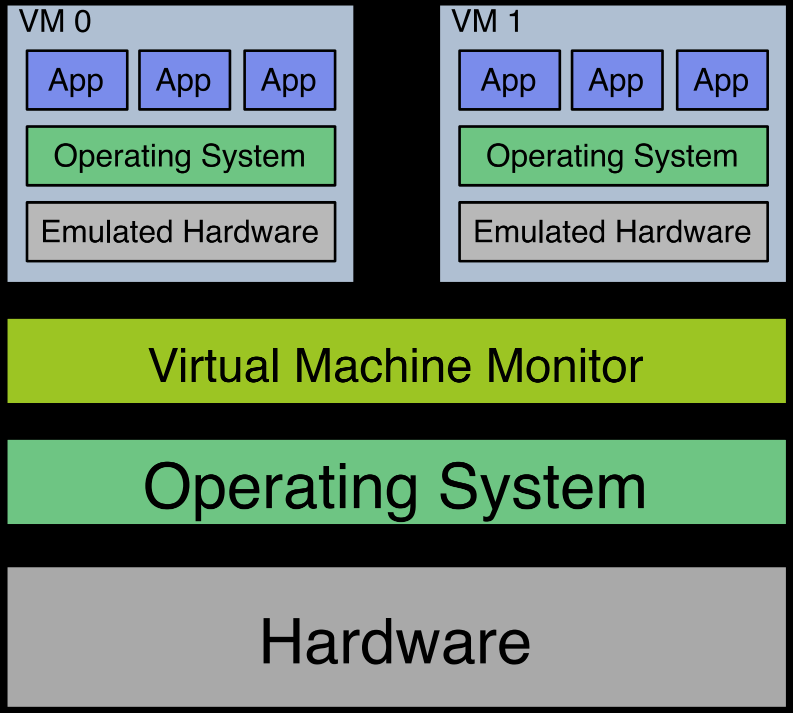
- Modern operating systems perform direct device access in kernel
- “Virtualize” CPU time and devices to applications
  - Pre-emptive multitasking
  - Hardware abstractions



project CHINASHOP

Software-Based Virtualization

- Run multiple operating systems concurrently
- Software Virtual Machine Monitor (VMM) virtualizes hardware
- Approaches:
  - Instruction Interpretation and translation

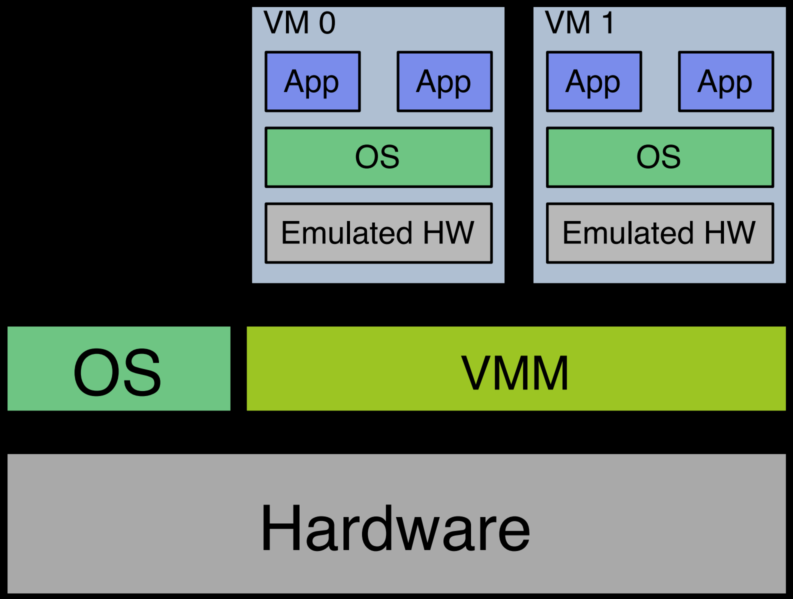


Interpretation and Translation

- Interpret processor instructions individually
  - Used if virtual machine may not be the same architecture as the host
- Translate and cache instruction fragments
  - Translate instructions to native instruction set and execute that instead
- Translate privileged instructions
  - Run user mode code natively
  - Translate privileged instructions to emulate expected behavior

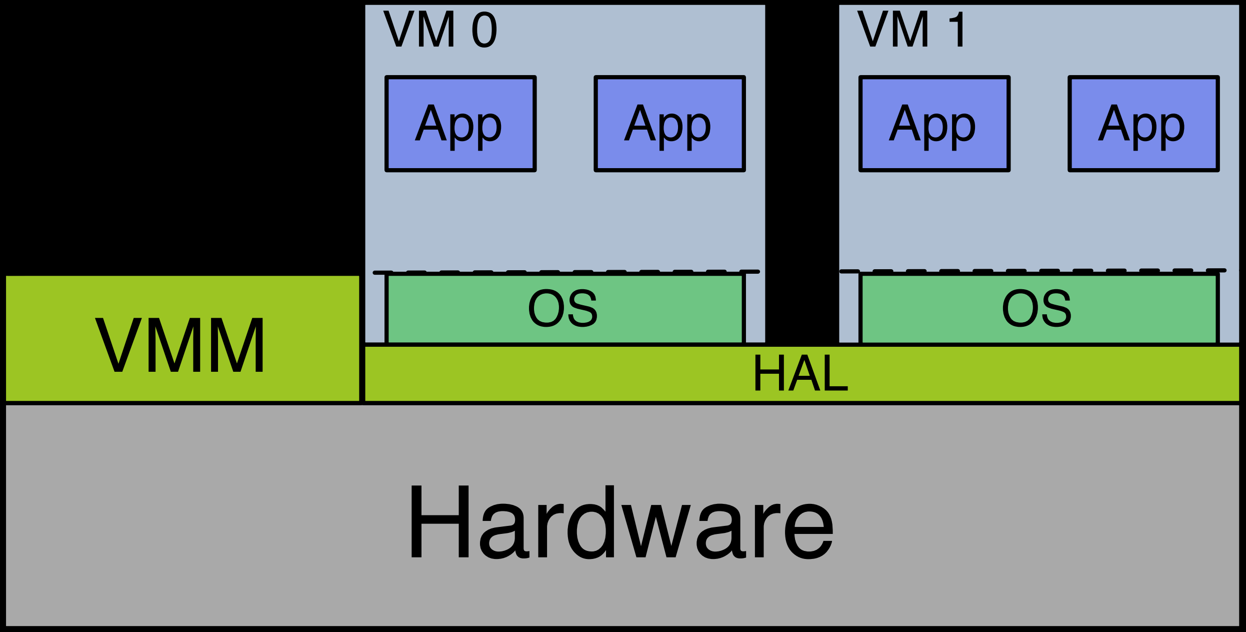
VMware

- VMM occupies Ring 0 along with Host and Guest OS
- Guest kernel code is translated
- Guest user code runs in ring 3
- Host memory is not mapped in guest
- VMM memory is protected from guest



project CHINASHOP

Hardware Virtualization



Hardware Virtualization

- Abstracts CPU beyond Ring 0 or Supervisor mode
- New VMM instructions can only be issued in “root” domain
- Events cause transition from guest OS to hypervisor OS.
- Guest/Host state is stored in memory

Hardware Virtualization

- IBM Logical Partitioning (LPAR)
  - IBM POWER5 processors (1999)
- Intel VT
  - VT-I: Future Itanium processors
  - VT-x: Core Duo and Solo (Jan 2006)
- AMD Pacifica
  - Athlon 64 X2 and FX (June 2006)

Intel Virtual Machine Extensions

Intel VT-x Overview

- Processor operates in two different modes

  VMX root (fully privileged ring 0)

  VMX non-root (less privileged ring 0)

- Virtual Machine Monitor launches Virtual Machines in VMX non-root mode
- Events may cause a *VM exit*
  - Selective exceptions, I/O device access, instructions, special register access
  - VMX non-root state is swapped out
  - VMX root state is swapped in

Intel VT-x in Detail

- Adds 10 new instructions
- Stores host and guest state in Virtual Machine Control Structure (VMCS)
  - Control registers
  - Debug register (DR7)
  - RSP, RIP, RFLAGS
  - Selector, base, limit, and access rights for segments (CS, SS, DS, ES, FS, GS, LDTR, TR)
  - GDTR, IDTR limit and base
  - MSRs

VMX Instruction Set

VMXON/VMXOFF	Enable/Disable VMX operation
VMCLEAR	Initialize VMCS region
VMPTRLD/VMPTRST	Load/Store Current VMCS pointer
VMREAD/VMWRITE	Read or Write VMCS fields
VMLAUNCH/VMRESUME	Launch or resume virtual machine
VMCALL	Issued from virtual machine to call into VMM

Interesting things about VT-x

- The entire OS-visible state of the processor is swapped in/out of memory
- Virtual Machines can have direct memory and device access
  - Intended to minimize VM exit overhead
  - Direct access to portions of I/O space or memory can be trapped
- Preventing detection was a design goal:
  - "There is no software-visible bit whose setting indicates whether a logical processor is in VMX non-root operation. This fact may allow a VMM to prevent guest software from determining that it is running in a virtual machine" -- Intel VT-x specification

Potential VT-x Hacks

- Run native OS as VM, use VT-x for:
  - Fast sleep and resume
  - Remote kernel debugging
  - “Safe-mode” driver development
    - *Checkpoint OS state before entering development driver*
    - *Resume from checkpoint if there is a fault*
    - *Remote debugging is a pain*
- Really nasty rootkits

Vitriol: The VT-x Rootkit

project CHINASHOP

Virtual Machine Rootkits

- *SubVirt*, Samuel T. King et al, University of Michigan and Microsoft Research
  - Malicious kernel module modifies boot sequence to load original OS inside Virtual PC
- *Vitriol*, Dino Dai Zovi, Matasano Security
  - VM rootkit for MacOS X using Intel VT-x on Intel Core Duo/Solo
- *BluePill*, Joanna Rutkowska, COSEINC
  - VM rootkit for Windows Vista x64 using AMD Pacifica on AMD Athlon 64

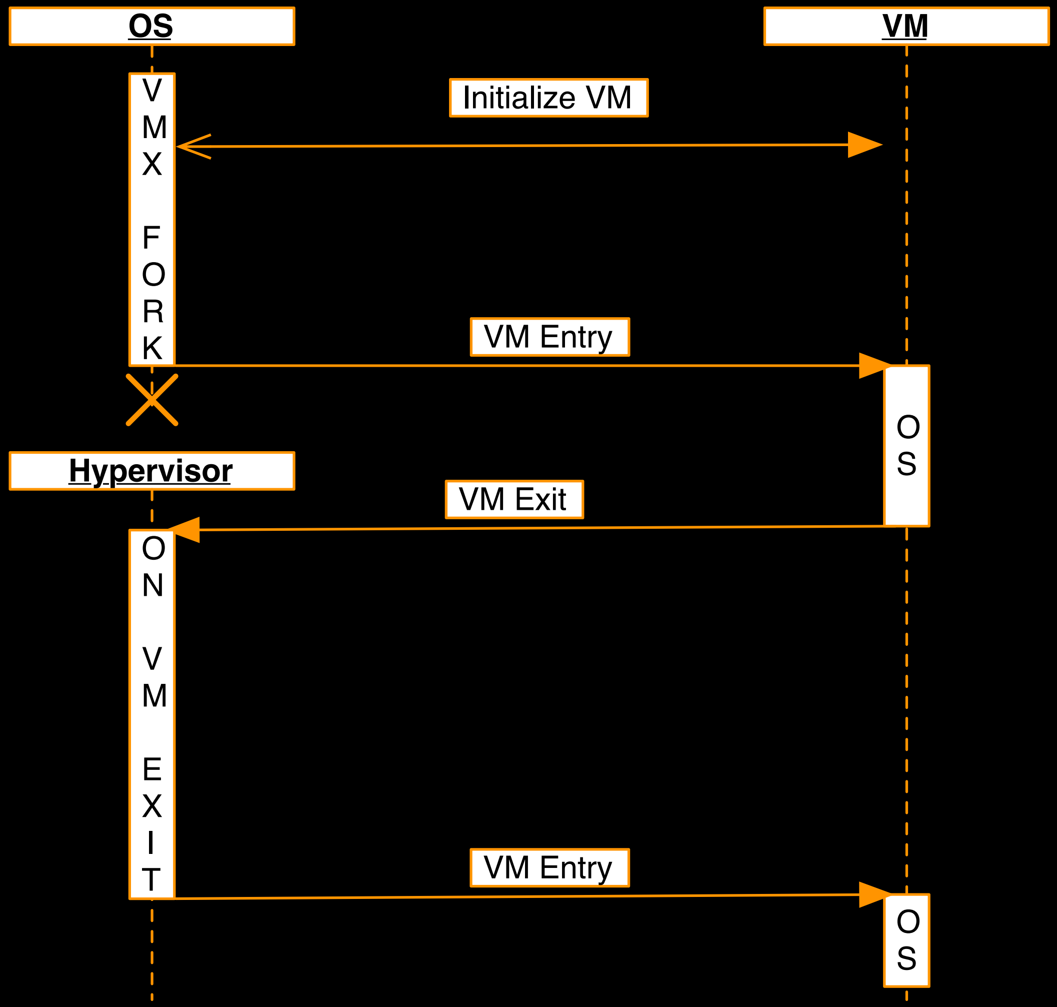
Hardware VM Rootkits

- Starts running in kernel in ring 0, installs *rootkit hypervisor*.
- Carves out some memory for hypervisor
- Migrates running OS into a VM
- Intercepts access to selected hardware devices
- Responds to “magic” instructions

Implementing a MacOS X VT-x Rootkit

- Loadable Kernel Extension installs rootkit and unloads itself
- Three main functions:
  - Vmx_init()
    - *Detects and initializes VT-x capabilities*
  - Vmx_fork()
    - *Migrate running OS into VM, fork running system into Guest VM and Host hypervisor*
  - On_vm_exit()
    - *Handle VM exit events*

VM Launch Sequence



project CHINASHOP

vmx_init()

- Check for VMX in CPUID and feature control MSR
- Enable VMX in CR4
- Allocate physical memory page for Virtual Machine Control Store (VMCS)
- Enable VMX operation for current processor with VMXON instruction
  - VMX operation and state is per-processor
  - You must lock your kernel thread to one processor

vmx_fork()

- Allocate code, stack, data for hypervisor
- Migrates running operating system into VM
- Set VM state to current state of running OS
- Set execution controls to minimize VM exits
    - Ignore guest exceptions, IO access, etc.
- Execution in VM continues running OS
- On VM exits, rootkit hypervisor executes

on_vm_exit()

- Handles VM exit events
- Emulate expected behavior for instructions like CPUID, CR0-CR4 access, RDMSR/WRMSR, etc.
- Implements backdoor functionality
  - CPUID instruction command channel
  - Filter/monitor/record device access
  - Hide blocks on disk by filtering ATAPI packets
  - Record keystrokes

CPUID Command Channel

- CPUID always causes a VM Exit
- CPUID can be executed in ring 3
- Magic values in EAX indicate requested action
- Action performed on running OS or value returned in registers
  - Change UID of specified process to 0 (root)
  - Hide specified process

Challenges

- VMX operation is per-CPU, keeping kernel thread on one CPU is tough
- Migrating one CPU or core of SMP system into VM might be tricky
- Observing raw device access requires mini-drivers to decode ATAPI/USB packets, etc.

Detecting VT-x Rootkits

- There is no hardware bit or register that indicates that the processor is running in VMX non-root mode
- Approaches:
  - Attempt to use VMX to create a VM
  - Attempt to detect latency caused by VM exit events

The VMX Test

- VMX instructions always cause a VM exit
- Create a simple VM to execute a few arithmetic instructions and store result
- If a host should support VMX, but it fails, host may be in a VM
- Is a rootkit going to fully emulate VMX?

VM Exit Latency

- Some instructions always cause VM Exit:
  - CPUID, INVD, MOV from CR3, RDMSR, WRMSR and VMX instructions
- Measure latency of these instructions using RDTSC

Latencies on Core Duo 2.16

Instruction	VMX Root	VMX Non-Root
ENTER/LEAVE	~14	~14
CPUID	~200	~3000

Countering Latency Measurements

- VT-x supports TSC offset for guests
- On a VM exit, get current TSC
- Before VM re-launch, add elapsed TSC to guest negative TSC offset
- Guest may still be able to detect clock skew against "real world" time

Demonstration

project CHINASHOP

Future Work

- Support multiple cores
  - Yes, I cheat and turn off one of my cores
- Manipulate VM's page table to hide rootkit pages
- Implement remote access features
  - Requires a good way to hook functions in the virtualized OS…

For More Information…

- Rootkit or source code is not available
- Xen 3.0 source code
- “Subverting the Windows Kernel for Fun and Profit”, Joanna Rutkowska
  - Discusses her AMD Pacifica Rootkit for Windows Vista x64

matasano

Question the answers. But not my answers to your questions.